REST, JPA, JSON and DTO's

Exercise 0: JPQL

1. Create a database jpql-demo
2. Git clone this project:https://github.com/dat3-codesamples-fall2020/jpql-demo-for-day4.git
3. Solve the small exercises in the Tester.java class

This should be very simple, you will find the solution to all but the last at the start of this document:
https://en.wikibooks.org/wiki/Java_Persistence/JPQL
It's only meant to give you the background for the JPQL-related parts of the next exercise

Exercise 1: Using the start code

1. Find the startcode on moodle and git clone it. Use this project for REST and JPA projects for the remainings of flow-1
    - You might have to setup your project server (right-click project → properties → run: Set server to your tomcat server)
2. Create a new database on your virtual Docker MySQL instance called `Week1Day4`
3. Change the Persistence Unit file to connect to your new database
4. Create a new JPA Entity: Employee with these fields (add constructors + getters/setters)
   Id
   Name
   Address
   Salary
5. Create an Employee facade class that can:
    - `getEmployeeById`
    - `getEmployeesByName`
    - `getAllEmployees`
    - `getEmployeesWithHighestSalary`
    - `createEmployee`

   Hint: Use the suggested facade architecture given in this exercise:
   https://docs.google.com/document/d/1Uib8GtBXmQZJ9x5tqXXHt1UYkkRPo9zKwugWa87bzUI/edit?usp=sharing

6. Create a JUnit test for all 5 methods

Exercise 2: Create DTO class

1. Create a new package called: dto
2. Inside it create a new java class: EmployeeDTO
3. Create only id, name and address fields and create a constructor that takes an Employee object as argument and sets data for its own fields.

Exercise 3: Exposing endpoints

1. Create a new web service to expose the facade methods:
2. Create the following endpoints using EmployeeDTO to wrap around the Employee entities.
    - `/api/employee/all`
    - `/api/employee/{id}`
    - `/api/employee/highestpaid`
    - `/api/employee/name/{name}`
3. Test the endpoints in a browser

Exercise 4: Deploy

1. Build project to create .war file
2. Use the tomcat manager on your digital ocean droplet to deploy your .war file

ssh into your droplet using an ssh-terminal (git-bash on Windows). Get into the tomcat containers shared volume: cd tomcat_mysql_nginx_docker/tomcat/webapps/jpa_rest_startup-1.0/WEB-INF/classes/META-INF
Now edit the persistence.xml file to change the database credentials from dev:ax2 to whatever you made your mysql user credentials on the droplet mysql container.

Test that your application works in the browser using the IP of your droplet.